Efficiently Deciding Uniqueness of Normal Forms and Unique Normalization for Ground TRSs*

Bertram Felgenhauer

University of Innsbruck
bertram.felgenhauer@uibk.ac.at

Abstract

We present an almost linear time algorithm for deciding uniqueness of normal forms for ground TRSs, and a cubic time algorithm for deciding unique normalization for ground TRSs.

1 Introduction

It is known that $\mathsf{UN}^{=}$ and $\mathsf{UN}^{\to}$ are decidable in polynomial time for ground TRSs. In this note, we are interested in bounding the exponent of the polynomial, which is of great interest to implementers. As far as we know, the best previous result for $\mathsf{UN}^{=}$ is an almost quadratic algorithm by Verma et al. [9] with $O(||\mathcal{R}||^2 \log ||\mathcal{R}||)$ time complexity, where $||\mathcal{R}||$ denotes the sum of the sizes of the sides of $\mathcal{R}$. In Section 3 we present an algorithm that decides $\mathsf{UN}^{=}$ in $O(||\mathcal{R}|| \log ||\mathcal{R}||$ time. In fact our algorithm is closely related to another algorithm by Verma [8, Theorem 31], but some care is needed to achieve an almost linear bound.

In the case of $\mathsf{UN}^{\to}$ for ground TRSs, Verma [8] and Godoy and Jacquemard [4] have established that polynomial time algorithms exist, using tree automata techniques. No precise bound is given by these authors. In Section 4 we will sketch (due to limited space) an $O(||\mathcal{R}||^3)$ time algorithm for deciding $\mathsf{UN}^{\to}$, based on a rewriting analysis reminiscent of the cubic time algorithm for confluence in [3].

2 Preliminaries

We assume familiarity with term rewriting and (bottom-up) tree automata, see [1, 2]. Fix a finite signature Σ. A tree automaton $\mathcal{A} = (Q, Q_f, \Delta)$ consists of a finite set of states Q disjoint from Σ, a set of final states $Q_f \subseteq Q$, and a set Δ of transitions $f(q_1, \ldots, q_n) \to q$ and ϵ-transitions $p \to q$, where f is an n-ary function symbol and $q_1, \ldots, q_n, p, q \in Q$. A deterministic tree automaton is an automaton without ϵ-transitions whose transitions have distinct left-hand sides (we do not require deterministic tree automata to be completely defined). Note that Δ can be viewed as a ground TRS over an extended signature that contains Q as constants. We write $\to_{\mathcal{A}}$ for $\to_{\Delta}$, where we regard the transitions as rewrite rules. For a TRS $\mathcal{R}$ we define $\mathcal{R}^{-} = \{r \to \ell \mid \ell \to r \in \mathcal{R}\}$. We write $t \trianglelefteq \mathcal{R}$ if t is a subterm of a side of a rule in $\mathcal{R}$. The size $||\mathcal{R}||$ of $\mathcal{R}$ is the sum of the sizes of the sides of $\mathcal{R}$. The unique normal forms property (convertible normal forms are equal) and the unique normalization property (no term reaches two distinct normal forms) are denoted by $\mathsf{UN}^{=}$ and $\mathsf{UN}^{\to}$, respectively. For a relation $\to$, $\to^{\shortparallel}$ denotes its parallel closure, $\to^{!}$ denotes reduction to a normal form, and $s{\downarrow}$ denotes a normal form of s. In particular, $s \to^{!} s{\downarrow}$.

*This research was supported by FWF project P27528.

3 Deciding $\mathsf{UN}^{=}$

We need some preparation before deciding $\mathsf{UN}^{=}$.

3.1 Currying

Currying allows us to turn an arbitrary TRS into one over constants and a single binary function symbol, thereby bounding the maximum arity of the resulting TRS.

In order to curry a TRS $\mathcal{R}$, we change all function symbols in Σ to be constants, and add a fresh, binary function symbol $\circ$, which we write as a left-associative infix operator. We define

$$(f(t_1, \ldots, t_n))^\circ = f \circ t_1^\circ \circ \cdots \circ t_n^\circ$$

The curried version of $\mathcal{R}$ is given by $\mathcal{R}^\circ = \{\ell^\circ \to r^\circ \mid \ell \to r \in \mathcal{R}\}$.

For ground systems, currying reflects and preserves $\mathsf{UN}^{\to}$ and $\mathsf{UN}^{=}$. For reflection, a direct simulation argument works ($s \to_\mathcal{R} t$ implies $s^\circ \to_{\mathcal{R}^\circ} t^\circ$, and s° is a $\mathcal{R}^\circ$-normal form if and only if s is an $\mathcal{R}$-normal form). For preservation, Kenneway et al. [5] show that $\mathsf{UN}^{\to}$ is preserved by currying for left-linear systems, and that $\mathsf{UN}^{=}$ is preserved by currying for arbitrary TRSs.

3.2 Recognizing Normal Forms

With $Q = Q_f = \{[s] \mid s \trianglelefteq \mathcal{R}\}$ and $\Delta = \{f([s_1], \ldots, [s_n]) \to [f(s_1, \ldots, s_n)] \mid f(s_1, \ldots, s_n) \trianglelefteq \mathcal{R}\}$ we obtain a deterministic tree automaton that accepts the subterms of $\mathcal{R}$. We modify this automaton to recognize normal forms. To this end, let $\star$ be a fresh constant and let

$$\begin{aligned}
Q' = Q'_f &= \{[s] \mid s \trianglelefteq \mathcal{R} \text{ and } s \text{ is } \mathcal{R}\text{-normal form}\} \cup \{[\star]\} \\
\Delta' &= \{f([s_1], \ldots, [s_n]) \to [f(s_1, \ldots, s_n)] \mid [f(s_1, \ldots, s_n)] \in Q'_f\} \cup \\
&\quad \{f([s_1], \ldots, [s_n]) \to [\star] \mid f \in \Sigma,\ [s_1], \ldots, [s_n] \in Q'_f,\ f(s_1, \ldots, s_n) \not\trianglelefteq \mathcal{R}\}
\end{aligned}$$

The state $[\star]$ accepts those $\mathcal{R}$-normal forms that are not subterms of $\mathcal{R}$.

Proposition 1. *The automaton $\mathcal{N}_\mathcal{R} = (Q', Q'_f, \Delta')$ recognizes the $\mathcal{R}$-normal forms over Σ.* □

3.3 Congruence Closure

Congruence closure (introduced by Nelson and Oppen [6]; a clean and fast implementation can be found in [7]) is an efficient method for deciding convertibility of ground terms modulo a set of ground equations $\mathcal{R}$.

The congruence closure consists of two phases. In the first phase, the procedure determines the congruence classes (hence the name) among the subterms of the given set of equations, where two subterms s and t are identified if and only if they are convertible, $s \leftrightarrow_\mathcal{R}^* t$. We write $[s]_\mathcal{R}$ for the convertibility class of s. In the second phase, given two terms u and v, we compute the normal forms with respect to rules

$$\mathcal{C} = \{f([s_1]_\mathcal{R}, \ldots, [s_n]_\mathcal{R}) \to [f(s_1, \ldots, s_n)]_\mathcal{R} \mid f(s_1, \ldots, s_n) \trianglelefteq \mathcal{R}\}$$

The terms u and v are $\mathcal{R}$-convertible if and only if $u{\downarrow_\mathcal{C}} = v{\downarrow_\mathcal{C}}$. We observe the following.

Proposition 2. *If we regard $[s]_\mathcal{R}$ for subterms $s \trianglelefteq \mathcal{R}$ as fresh constants, the set $\mathcal{C}$ is an orthogonal, ground TRS whose rules, as transitions of a tree automaton, are deterministic.* □

```
1: compute C_R and a representation of N_R
2: for all constants c ⊴ R that are normal forms do
3:     push ([c]_R, [c]) to worklist
4: while worklist not empty do
5:     (p, q) ← pop worklist
6:     if seen(p) is defined then
7:         return UN^=(R) is false
8:     seen(p) ← q
9:     for all transitions p_1 ∘ p_2 → p_r ∈ C_R with p ∈ {p_1, p_2} do
10:        if q_1 = seen(p_1) and q_2 = seen(p_2) are defined then
11:            if there is a transition q_1 ∘ q_2 → q_r ∈ N_R then
12:                push (p_r, q_r) to worklist
13: return UN^=(R) is true
```

Figure 1: Deciding $\mathsf{UN}^{=}(\mathcal{R})$

Consequently, we may represent $\mathcal{C}$ as a deterministic tree automaton $\mathcal{C}_\mathcal{R} = (Q, Q_f, \Delta)$ with $Q = Q_f = \{[s]_\mathcal{R} \mid s \trianglelefteq \mathcal{R}\}$ and $\Delta = \mathcal{C}$. Each state $[s]_\mathcal{R}$ accepts precisely the terms convertible to s. Note that the automaton is not completely defined in general: Only terms s that allow a conversion $s \leftrightarrow^*_\mathcal{R} t$ with a root step are accepted.

3.4 Checking $\mathsf{UN}^{=}$

Given a ground TRS $\mathcal{R}$, we want to decide $\mathsf{UN}^{=}(\mathcal{R})$, that is, whether any two $\mathcal{R}$-convertible $\mathcal{R}$-normal forms are equal.

First note that if we have two distinct convertible normal forms $s \leftrightarrow^*_\mathcal{R} t$ such that the conversion does not contain a root step, then there are strict subterms of s and t that are convertible and distinct. Therefore, $\mathsf{UN}^{=}(\mathcal{R})$ reduces to the question whether any state of $\mathcal{C}_\mathcal{R}$, the automaton produced by the congruence closure of $\mathcal{R}$, accepts more than one normal form. Let $\mathcal{C}_\mathcal{R} \cap \mathcal{N}_\mathcal{R}$ be the result of the product construction on $\mathcal{C}_\mathcal{R}$ and $\mathcal{N}_\mathcal{R}$. We can decide $\mathsf{UN}^{=}$ by enumerating accepting runs $t \to^*_{\mathcal{C}_\mathcal{R} \cap \mathcal{N}_\mathcal{R}} (q_1, q_2)$ in a bottom-up fashion until either

- we obtain two distinct accepting runs ending in (q_1, q_2) and (q_1', q_2') with $q_1 = q_2$, in which case $\mathsf{UN}^{=}(\mathcal{R})$ does not hold; or
- we have exhausted all runs, in which case $\mathsf{UN}^{=}(\mathcal{R})$ holds.

Assume that $\mathcal{R}$ is curried. The enumeration of accepting runs can be performed by the algorithm in Figure 1. The correctness of the procedure hinges on two key facts: First, the automaton $\mathcal{C}_\mathcal{R} \cap \mathcal{N}_\mathcal{R}$ is deterministic, which means that distinct runs result from distinct terms. Secondly, the set of $\mathcal{R}$ normal forms is closed under subterms, so we can skip non-normal forms in the enumeration.

Theorem 3. *The algorithm in Figure 1 is correct and runs in $O(||\mathcal{R}|| \log ||\mathcal{R}||)$ time.*

Proof. We have already argued correctness, so let us focus on the complexity. Let $n = ||\mathcal{R}||$. First we compute $\mathcal{C}_\mathcal{R}$ using the congruence closure algorithm from [7] in $O(n \log n)$ time. While $\mathcal{N}_\mathcal{R}$ has quadratically many transitions, we can define the transitions as a partial function using $O(n \log n)$ time for preparation and $O(\log n)$ time per invocation of the transition function. This bound relies on currying, for constant size left-hand sides, and on perfect sharing of terms, for $O(\log n)$ subterm tests. This covers line 1 of the algorithm. Lines 2 to 3 take $O(n)$ time. Note

that lines 8 to 12 are executed at most once per state of $\mathcal{C}_\mathcal{R}$, i.e., $O(n)$ times. The enumeration on line 9 can be precomputed in $O(n)$ time, by creating an array of lists of transitions indexed by the states of $\mathcal{C}_\mathcal{R}$ and adding each transition $q_1 \circ q_2 \to q_r \in \mathcal{C}_\mathcal{R}$ to the lists indexed by q_1 and q_2 (if $q_1 \neq q_2$). Because each transition is added to at most two lists, lines 10 to 12 are executed at most twice per transition in $\mathcal{C}_\mathcal{R}$, so $O(n)$ times. The check on line 11 takes $O(\log n)$ time per iteration, so $O(n \log n)$ time in total. Finally, we note that line 12 is executed $O(n)$ times, so no more than $O(n)$ items are ever added to the worklist, which means that lines 4 to 7 are executed $O(n)$ times. Overall the algorithm executes in $O(n \log n)$ time, as claimed. □

4 Deciding $\mathsf{UN}^{\to}$

4.1 Preparation: Flattening, Rewrite Closure, Meetable Constants

To simplify the reachability analysis in the $\mathsf{UN}^{\to}$ property, we flatten the ground TRS $\mathcal{R}$, which we assume to be curried. To this end, we add fresh constants $[s]$ for $s \trianglelefteq \mathcal{R}$, and take the rules

$$\mathcal{E} = \{f([s_1], \ldots, [s_n]) \to [f(s_1, \ldots, s_n)] \mid f(s_1, \ldots, s_n) \trianglelefteq \mathcal{R}\}$$

The flattened TRS is $\mathcal{R}' = \{[\ell] \to [r] \mid \ell \to r \in \mathcal{R}\} \cup \mathcal{E}$. This system simulates rewriting by $\mathcal{R}$.

Proposition 4. ${}_{\mathcal{E}^-}{}^{!}\!\leftarrow \cdot \to_{\mathcal{R}'} \cdot \to^{!}_{\mathcal{E}^-} \subseteq \to_\mathcal{R} \subseteq \to^*_\mathcal{E} \cdot \to_{\mathcal{R}'} \cdot {}^{*}_{\mathcal{E}}\!\leftarrow$.

In the following, p and q range over $[t]$ with $t \trianglelefteq \mathcal{R}$. Following [3, Section 3.2], we define the rewrite closure $\mathcal{F}$ of $\mathcal{R}'$ inductively by the following inference rules:

$$\frac{t \trianglelefteq \mathcal{R}}{[t] \to [t] \in \mathcal{F}}\ \text{refl} \qquad \frac{p_1 \circ p_2 \to p \in \mathcal{E} \quad p_1 \to q_1 \in \mathcal{F} \quad p_2 \to q_2 \in \mathcal{F} \quad q_1 \circ q_2 \to q \in \mathcal{E}}{p \to q \in \mathcal{F}}\ \text{comp}$$

$$\frac{p \to q \in \mathcal{R}'}{p \to q \in \mathcal{F}}\ \text{base} \qquad \frac{p \to q \in \mathcal{F} \quad q \to r \in \mathcal{F}}{p \to r \in \mathcal{F}}\ \text{trans}$$

Proposition 5 ([3, Lemma 3.4])**.** $\to^*_\mathcal{R} \subseteq \to^*_{\mathcal{E}\cup\mathcal{F}} \cdot {}_{\mathcal{E}\cup\mathcal{F}^-}{}^{*}\!\leftarrow$. □

We say two constants p and q are meetable if $p\ {}_{\mathcal{E}\cup\mathcal{F}}{}^{*}\!\leftarrow \cdot \to^*_{\mathcal{E}\cup\mathcal{F}} q$. In this case we write $p \uparrow q$. The relation $\uparrow$ is dual to $\downarrow$ in [3, Section 3.5] and can be computed as follows.

$$\frac{t \trianglelefteq \mathcal{R}}{[t] \uparrow [t]}\ \text{refl} \qquad \frac{p_1 \circ p_2 \to p \in \mathcal{E} \quad p_1 \uparrow q_1 \quad p_2 \uparrow q_2 \quad q_1 \circ q_2 \to q \in \mathcal{E}}{p \uparrow q}\ \text{comp}$$

$$\frac{q \to p \in \mathcal{F} \quad q \uparrow r}{p \uparrow r}\ \text{trans}_l \qquad \frac{p \uparrow q \quad q \to r \in \mathcal{F}}{p \uparrow r}\ \text{trans}_r$$

4.2 Peak Analysis

Using the rewrite closure, any peak $s\ {}^{*}_{\mathcal{R}}\!\leftarrow \cdot \to^*_\mathcal{R} t$ between normal forms s and t can be decomposed as

$$s \xrightarrow[\mathcal{E}\cup\mathcal{F}^-]{*} \cdot \xleftarrow[\mathcal{E}\cup\mathcal{F}]{*} \cdot \xrightarrow[\mathcal{E}\cup\mathcal{F}]{*} \cdot \xleftarrow[\mathcal{E}\cup\mathcal{F}^-]{*} t$$

If s and t are chosen to be of minimal size, then there must be a root step. Hence, without loss of generality, there is a constant q such that

$$s \xrightarrow[\mathcal{E}\cup\mathcal{F}^-]{*} q \xleftarrow[\mathcal{E}\cup\mathcal{F}]{*} \cdot \xrightarrow[\mathcal{E}\cup\mathcal{F}]{*} \cdot \xleftarrow[\mathcal{E}\cup\mathcal{F}^-]{*} t \tag{1}$$

Note the special case $s \to^*_{\mathcal{E}\cup\mathcal{F}^-} q\ {}_{\mathcal{E}\cup\mathcal{F}^-}{}^{*}\!\leftarrow t$, which implies that any q is reachable from at most one normal form using rules from $\mathcal{E} \cup \mathcal{F}^-$.

4.3 Checking $\mathsf{UN}^{\to}$

The computation consists of several steps. Using the relation $\uparrow$, (1) becomes

$$s \xrightarrow[\mathcal{E}\cup\mathcal{F}^-]{*} q \xleftarrow[\mathcal{E}\cup\mathcal{F}]{*} C[q_1,\ldots,q_n] \uparrow^{\shortparallel} C[p_1,\ldots,p_n] \xleftarrow[\mathcal{E}\cup\mathcal{F}^-]{*} t \tag{2}$$

First, we compute the partial function $w(q)$ that maps q to the normal form s with $s \to^*_{\mathcal{E}\cup\mathcal{F}^-} q$, the first part of the conversion (2). If any q is reachable from more than one normal form, $\mathsf{UN}^{\to}$ does not hold. To perform this computation efficiently, we make use of the automaton $\mathcal{N}_\mathcal{R}$ that recognizes normal forms. The code is similar to Figure 1, lines 2 to 13, but using the automaton $\mathcal{A} = (Q, Q_f, \Delta)$ given by $Q = Q_f = \{[s] \mid s \trianglelefteq \mathcal{R}\}$ and $\Delta = \mathcal{E} \cup \mathcal{F}^-$ instead of $\mathcal{C}_\mathcal{R}$. Because the product automaton is no longer deterministic, we actually have to compute witnesses and only fail in line 7 if the witnesses are different. Furthermore, in addition to lines 9 to 12, we need a similar loop processing the ϵ-transitions (from $\mathcal{F}^-$). The latter change increases the complexity from $O(||\mathcal{R}|| \log ||\mathcal{R}||$ to $O(||\mathcal{R}||^2)$.

Secondly, we analyze the right part of the conversion (2). To this end, we compute the partial function $w'(q)$ that maps q to the normal form t with $q \;{}^{*}_{\mathcal{E}\cup\mathcal{F}}\!\!\leftarrow \cdot \uparrow^{\shortparallel} \cdot \;{}^{*}_{\mathcal{E}\cup\mathcal{F}^-}\!\!\leftarrow t$, or to ∞ if there is more than one such normal form. Note that by (2) with $C = \Box$, we have $w(p) = w'(q)$ or $w'(q) = \infty$ whenever $p \uparrow q$ and $w(q)$ is defined. Extending the base cases to larger contexts requires analyzing the $q \;{}^{*}_{\mathcal{E}\cup\mathcal{F}}\!\!\leftarrow C[q_1,\ldots,q_n]$ sequence and can be done almost the same way as the computation of $w(q)$, using $\mathcal{E} \cup \mathcal{F}$ instead of $\mathcal{E} \cup \mathcal{F}^-$. The complexity of this computation is still $O(||\mathcal{R}||^2)$, despite a subtlety: whereas $w(q)$ is updated at most once, each $w'(q)$ may be updated twice: to record a witness, and to record that there is more than one witness.

The system has the $\mathsf{UN}^{\to}$ property if $w'(q) = w(q)$ whenever $w(q)$ is defined. Overall, the computation is dominated by the computation of the rewrite closure and the meetable constants, which take $O(||\mathcal{R}||^3)$ time [3]. Hence, $\mathsf{UN}^{\to}$ can be decided in cubic time.

References

[1] F. Baader and T. Nipkow. *Term Rewriting and All That.* Cambridge University Press, 1998.

[2] H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, C. Löding, S. Tison, and M. Tommasi. *Tree Automata Techniques and Applications.* 2007. URL: http://tata.gforge.inria.fr.

[3] B. Felgenhauer. Deciding confluence of ground term rewrite systems in cubic time. In *Proc. 23rd International Conference on Rewriting Techniques and Applications*, volume 15 of *Leibniz International Proceedings in Informatics*, pages 165–175, 2012.

[4] G. Godoy and F. Jacquemard. Unique normalization for shallow TRS. In *Proc. 20th International Conference on Rewriting Techniques and Applications*, volume 5595 of *Lecture Notes in Computer Science*, pages 63–77, 2009.

[5] R. Kennaway, J.W. Klop, M. Ronan Sleep, and F.-J. de Vries. Comparing curried and uncurried rewriting. *Journal of Symbolic Computation*, 21(1):15–39, 1996.

[6] G. Nelson and D.C. Oppen. Fast decision procedures based on congruence closure. *Journal of the ACM*, 27(2):356–364, 1980.

[7] R. Nieuwenhuis and A. Oliveras. Fast congruence closure and extensions. *Information and Computation*, 205(4):557–580, 2007.

[8] R. Verma. Complexity of normal form properties and reductions for term rewriting problems. *Fundamenta Informaticae*, 92(1–2):145–168, 2009.

[9] R.M. Verma, M. Rusinowitch, and D. Lugiez. Algorithms and reductions for rewriting problems. *Fundamenta Informaticae*, 46(3):257–276, 2001.